

Práctica 4 - Diseño de ISA

Organización del Computador 1

Segundo Cuatrimestre 2014

Ejercicio 1 Dada una arquitectura con palabras de 32 bits ; diga cuántos *bits* son necesarios para especificar una dirección de memoria en los siguientes casos:

- a) Tamaño de memoria física: 4GB ; con direccionamiento a *byte*.
- b) Tamaño de memoria física: 8GB ; con direccionamiento a “*medias palabras*”.
- c) Tamaño de memoria física: 16GB ; con direccionamiento a *palabra*.
- d) Tamaño de memoria física: 32GB ; con direccionamiento a “*palabra doble*”.

Ejercicio 2 Dada una arquitectura con palabras e instrucciones de $b\text{ bytes}$ que trabaja con una memoria física de $x\text{ bytes}$, con direccionamiento a *palabra*:

- a) ¿Cuántos *bits* serán necesarios para especificar una dirección cualquiera de la memoria?
- b) ¿Cuál sería el número máximo de códigos de operación posibles suponiendo que todas las instrucciones incluyen sólo un operando con modo de direccionamiento directo a memoria?
- c) ¿Cómo reescribiría las respuestas a las dos preguntas anteriores si x y b fuesen potencias de dos (i.e., $x = 2^k$ y $b = 2^j$)?
- d) ¿Bajo que condiciones de j se obtiene la máxima granularidad de acceso a la memoria?

Ejercicio 3 ¿Cuál es el máximo número de instrucciones de una dirección que admitirían cada una de las siguientes máquinas?

- a) Instrucciones de 12 bits , direcciones de 4 bits , y 6 instrucciones de dos direcciones.
- b) Instrucciones de 16 bits , direcciones de 6 bits y n instrucciones de dos direcciones.

Ejercicio 4 Dada una máquina con instrucciones de 16 bits y direcciones de 4 bits , diseñe un formato de instrucción que contenga:

- I. 15 instrucciones de 3 direcciones;
- II. 14 instrucciones de 2 direcciones;
- III. 31 instrucciones de 1 dirección;
- IV. 16 instrucciones sin direcciones.

Ejercicio 5 Diseñe un formato de instrucción con código de operación extensible (la cantidad de *bits* del código de operación es variable) que se pueda codificar en una instrucción de 36 bits y permita lo siguiente:

- 7 instrucciones con dos direcciones de 15 bits y un número de registro de 3 bits ;

- 500 instrucciones con una dirección de 15 bits y un número de registro de 3 bits;
- 50 instrucciones sin direcciones ni registros.

Ejercicio 6 Suponiendo que se necesitan 3 bits para direccionar un registro, ¿es posible diseñar un formato de instrucción cuyo código de operación sea extensible y permita codificar lo siguiente en una instrucción de 12 bits?

- 4 instrucciones con tres registros;
- 255 instrucciones con un registro;
- 16 instrucciones sin registros.

Ejercicio 7 Sea un procesador que cuenta con un registro distinguido, llamado *acumulador*, sobre el que se realizan operaciones aritméticas; por ejemplo, sumar al acumulador un dato de memoria. El formato de las instrucciones que ejecuta es el siguiente:

4 bits	1 bit	7 bits
código de operación	md	dirección

Mediante el valor del bit *md* se escoge el modo de direccionamiento a utilizar: 1 para *directo* y 0 para *indirecto*.

Suponiendo que el código de la operación de suma mencionada más arriba es 1100, y el estado de la computadora es el indicado en la tabla, ¿Cuál será el contenido del acumulador después de ejecutar las siguientes instrucciones?

- 1100 1011 0100
- 1100 0011 0101

Acompañe el resultado con una explicación gráfica.

Memoria	
Dirección	Contenido
0x0034	0033h
0x0035	0036h
0x0036	0035h
0x0037	0034h
Registros	
Nombre	Contenido
Acumulador	2B58h

Ejercicio 8 ¹ Existen muchas razones por las que los diseñadores de computadoras quieren instrucciones de longitud fija. ¿Por qué no es una buena idea tener instrucciones de longitud fija en una máquina de pila?

Ejercicio 9 Dados cuatro modelos de computadoras con direcciones de 16 bits, códigos de operación de 8 bits y tamaño de instrucciones múltiplo de 4 bits; pero que pueden diferenciarse por las siguientes características:

Máquina 0: es una máquina *de pila*; sus instrucciones no precisan operandos (exceptuando PUSH y POP).

Máquina 0	
PUSH [M]	push [M]
POP [M]	[M] ← pop
ADD	push(pop+pop)
SUB	push(pop-pop)
MUL	push(pop*pop)
DIV	push(pop/pop)

Máquina 1: utiliza un registro *acumulador* (A) e instrucciones con un operando.

Máquina 1	
LOAD [M]	A ← [M]
STORE [M]	[M] ← A
ADD [M]	A ← A + [M]
SUB [M]	A ← A - [M]
MUL [M]	A ← A * [M]
DIV [M]	A ← A / [M]

La máquina 2 y la máquina 3 disponen de 16 registros e instrucciones de dos y tres operandos, que operan con cualquier combinación de direcciones de memoria y registros.

Máquina 2	
MOV X, Y	X ← Y
ADD X, Y, Z	X ← X + Y
SUB X, Y, Z	X ← X - Y
MUL X, Y, Z	X ← X * Y
DIV X, Y, Z	X ← X / Y

Máquina 3	
MOV X, Y	X ← Y
ADD X, Y, Z	X ← Y + Z
SUB X, Y, Z	X ← Y - Z
MUL X, Y, Z	X ← Y * Z
DIV X, Y, Z	X ← Y / Z

- Escriba, para cada máquina, el programa que computa $x = (a + b * c) / (d - e * f)$.

¹Ej. 6, Cap. 5 del L. Null & J. Lobur - Essentials Of Computer Organization And Architecture

- b) ¿Cuántas instrucciones tiene cada programa?
- c) Diseñe el formato de instrucción para cada uno.
- d) ¿Cuál es la longitud (en *bytes*) del programa para cada computadora?
- e) ¿Cuál es la cantidad de accesos a memoria en cada caso?

Ejercicio 10 Sea una arquitectura que posee 1024 registros de 64 *bits*. Cada registro se divide en cuatro partes de 16 *bits* que pueden ser referenciadas independientemente. La memoria física es de 2 *GB* direccionables a *byte*.

El procesador provee el repertorio de instrucciones listado a continuación; estas instrucciones se describen pensando a cada registro R_i como un vector de cuatro posiciones, referenciadas como $R_i(0)$, $R_i(1)$, $R_i(2)$ y $R_i(3)$.

- CLR $R_u(i)$ Pone en 0 la componente i de u .
- INT $R_u(i), R_v, R_w$ Almacena en la componente i de u el producto escalar entre v y w .
- VEC R_u, R_v, R_w Almacena en u el producto vectorial entre v y w .
- ADD R_u, R_v, R_w Almacena en u la suma componente a componente entre v y w .
- SDV $R_u, R_v, R_w(i)$ Almacena en u la división de cada componente de v por la componente i de w .
- MOV p, q Copia 64 *bits*; p y q pueden ser registros completos o direcciones de memoria.

- a) Diseñe un formato de instrucción de longitud fija de 64 *bits*.
- b) ¿Cuáles modos de direccionamiento emplea el formato de instrucción propuesto?
- c) ¿Qué longitud de palabra resultaría más adecuada para ese formato?

Ejercicio 11 Una arquitectura posee 32 registros de uso general de 32 *bits* (R_0 al R_{31}). La memoria es de 4 *GB*, direccionable a *byte*. La longitud de las palabras es de 32 *bits*. Los *flags* son Z (*zero*), N (*negative*), V (*overflow*) y C (*carry*).

El procesador soporta el siguiente conjunto de instrucciones, donde R_i representa al i -ésimo registro (con i entre 0 y 31), c representa una constante de 19 *bits* y si x representa a un entero n en complemento a 2 de 19 *bits*, $e32(x)$ representa a n en complemento a 2 de 32 *bits*.

AND R_i, R_j, R_k	$R_i \leftarrow R_j \text{ and } R_k$	JMP c	$PC \leftarrow PC + e32(c)$
OR R_i, R_j, R_k	$R_i \leftarrow R_j \text{ or } R_k$	JE c	$PC \leftarrow PC + e32(c)$ <i>sii</i> Z
XOR R_i, R_j, R_k	$R_i \leftarrow R_j \text{ xor } R_k$	JNE c	$PC \leftarrow PC + e32(c)$ <i>sii</i> not Z
ADD R_i, R_j, R_k	$R_i \leftarrow R_j + R_k$	JLE c	$PC \leftarrow PC + e32(c)$ <i>sii</i> Z or (N xor V)
SUB R_i, R_j, R_k	$R_i \leftarrow R_j - R_k$	JG c	$PC \leftarrow PC + e32(c)$ <i>sii</i> not (Z or (N xor V))
SADD R_i, R_j, c	$R_i \leftarrow R_j + e32(c)$	JL c	$PC \leftarrow PC + e32(c)$ <i>sii</i> N xor V
UADD R_i, R_j, c	$R_i \leftarrow R_j + c$	JGE c	$PC \leftarrow PC + e32(c)$ <i>sii</i> not (N xor V)
SSUB R_i, R_j, c	$R_i \leftarrow R_j - e32(c)$	JLEU c	$PC \leftarrow PC + e32(c)$ <i>sii</i> C or Z
USUB R_i, R_j, c	$R_i \leftarrow R_j - c$	JGU c	$PC \leftarrow PC + e32(c)$ <i>sii</i> not (C or Z)
GET $R_i, [R_j+c]$	$R_i \leftarrow [R_j + e32(c)]$	JC c	$PC \leftarrow PC + e32(c)$ <i>sii</i> C
SET $[R_i+c], R_j$	$[R_i + e32(c)] \leftarrow R_j$	JN c	$PC \leftarrow PC + e32(c)$ <i>sii</i> N
NOT R_i	$R_i \leftarrow \text{not } R_i$	JV c	$PC \leftarrow PC + e32(c)$ <i>sii</i> V
CLR R_i	$R_i \leftarrow 0$		

- a) Diseñar un formato de instrucción de longitud fija de 32 *bits* y constantes de 19 *bits*.
- b) ¿Podría escribir programas en lenguaje ensamblador usando sólo los saltos condicionales JMP, JE, JC, JN y JV para reemplazar a los otros saltos condicionales? En caso negativo exhibir un contra ejemplo, y en caso afirmativo justificar brevemente y exhibir algún ejemplo.

- c) Sin modificar la longitud de la instrucción, sugiera alguna nueva instrucción sobre el formato de instrucción propuesto de modo de facilitar al usuario de esta arquitectura cargar constantes de 32 bits en un registro. Escribir en una o más instrucciones la carga de una constante de 32 bits en un registro usando esa nueva instrucción.

Ejercicio 12 Dado un procesador con direcciones de 16 bits , 2 registros de uso general (R0 y R1), que soporte las siguientes instrucciones:

ADD R_i, R_j	$R_i \leftarrow R_i + R_j$
SUB R_i, R_j	$R_i \leftarrow R_i - R_j$
JZ c	$PC \leftarrow c$
JMP p	$PC \leftarrow p$
MOV q, r	$q \leftarrow r$

c es una constante de 16 bits ,

p puede ser R0 o una constante de 16 bits ,

q puede ser un registro o una dirección de memoria,

r puede ser una constante de 16 bits , un registro (distinto a q), o una dirección de memoria (si es que q no lo fue).

Las instrucciones son de 4 bits más las direcciones de memoria o valores inmediatos que requieran.

- Diseñar el formato de instrucción para este conjunto de instrucciones.
- ¿Cómo usaría el diseño anterior en caso que se necesite habilitar la posibilidad a una expansión del formato de instrucción sin la pérdida de la compatibilidad hacia atrás? Muestre al menos 5 nuevas instrucciones en el formato.

Ejercicio 13 En este ejercicio trabajaremos sobre una arquitectura diseñada para operar con números complejos (\mathbb{C}). Esta arquitectura posee 256 registros que miden 128 bits y cuyas mitades se denominan \mathcal{R} (parte real) e \mathcal{I} (parte imaginaria). De esta forma, si R8 denota al registro 8 completo, entonces $R8 = \{\mathcal{R}(R8); \mathcal{I}(R8)\}$ donde $\mathcal{R}(R8)$ e $\mathcal{I}(R8)$ referencian los 64 MSb (bits más significativos) y los 64 LSb (bits menos significativos) de R8 respectivamente. La memoria principal es de 16 GB , y su mínima unidad direccionable es de 64 bits . El procesador sólo soporta las siguientes 6 instrucciones:

CLR d	$d \leftarrow 0$	
MOV p, q	$p \leftarrow q$	(Copia 128 bits)
CNJ u, v	$u \leftarrow \bar{v}$	$u = \{\mathcal{R}(v); -\mathcal{I}(v)\}$
ADD u, v, w	$u \leftarrow v \oplus w$	$u = \{\mathcal{R}(v) + \mathcal{R}(w); \mathcal{I}(v) + \mathcal{I}(w)\}$
MUL u, v, w	$u \leftarrow v \otimes w$	$u = \{\mathcal{R}(v)\mathcal{R}(w) - \mathcal{I}(v)\mathcal{I}(w); \mathcal{R}(v)\mathcal{I}(w) + \mathcal{I}(v)\mathcal{R}(w)\}$
SDV u, v, w	$u \leftarrow v/\mathcal{R}(w)$	$u = \{\mathcal{R}(v)/\mathcal{R}(w); \mathcal{I}(v)/\mathcal{R}(w)\}$

Donde u, v, w son registros completos; p, q pueden ser registros completos de 128 bits o direcciones de memoria a partir de las cuales se almacenan 128 bits de datos; y d puede ser un registro completo o cualquiera de las mitades de un registro. Se pide:

- Diseñar un formato de instrucción de longitud fija de 64 bits .
- Indicar cuántos y cuáles modos de direccionamiento emplea el formato propuesto.
- Ensamblar (en hexadecimal) el siguiente programa empleando el formato propuesto:

```
MOV R111, [F0h]      ;R111 ← [0xF0]
CNJ R112, R111        ;R112 ←  $\overline{R111}$ 
MUL R111, R112, R111  ;R111 ←  $R112 \otimes R111$ 
SDV R111, R112, R111  ;R111 ←  $R112/\mathcal{R}(R111)$ 
MOV [F2h], R111       ;[0xF2] ← R111
```

- Suponiendo que a partir de la dirección de memoria 240 se encuentra almacenado cierto número complejo z (parte real de 64 bits seguida de parte imaginaria de 64 bits), escriba una expresión matemática para el valor que quedaría almacenado en la posición de memoria 242 después de ejecutarse el programa anterior.

Ejercicio 14 Considere una máquina muy sencilla, cuyo repertorio se muestra a continuación:

OPERACIÓN	EFFECTO	FLAG	OPCODE
ADD u, v	$[u] \leftarrow [u] + [v]$	$Z \Leftrightarrow ([u] + [v] = 0)$	00
TST u, v		$Z \Leftrightarrow ([u] = [v])$	01
MOV u, v	$[u] \leftarrow [v]$	$Z \Leftrightarrow ([v] = 0)$	10
BSS u	Si $Z = 1$, $PC \leftarrow u$	$Z \leftarrow 1$ (siempre)	11

donde u y v son direcciones de memoria (la arquitectura tiene direccionamiento a palabra), Z es el flag de cero, y PC el registro de próxima instrucción. La ALU trabaja en complemento a 2 de 16 bits. El formato de instrucción es el que sigue.

2 bits	7 bit	7 bits
código de operación	Dirección u	Dirección v ó 0000000

- ¿ Qué longitud de palabra considera más adecuada para la arquitectura ?
- Indicar cuánta memoria física podrá emplear la máquina como máximo.
- Suponiendo una longitud de palabra de 16 bits y que, a partir del *byte 0* de la memoria se encuentra almacenada la siguiente secuencia hexadecimal:

03 88 00 06 40 05 C5 80 C0 00 03 8B 00 01 03 88 00 06 40 05 03 8B

- Suponiendo que se tratase de un fragmento de programa, “desensamblar” el vuelco de memoria dado (es decir, escribirlo en lenguaje *assembler*).
- Considerando que inicialmente el PC está en 0 , explicar brevemente qué hace el fragmento. (Se espera aquí una explicación conceptual sencilla, por ejemplo: “El programa calcula la suma de los primeros siete números naturales, dejando el resultado en la dirección de memoria $0x4F$ ”).
- Decir cuántos accesos a memoria deberán ser realizados en total durante la ejecución del fragmento.
- Determinar cuántos valores diferentes llegarán a presentarse en el *bus* de direcciones de la máquina durante dicha ejecución.